Lab Class ML:I, ML:II

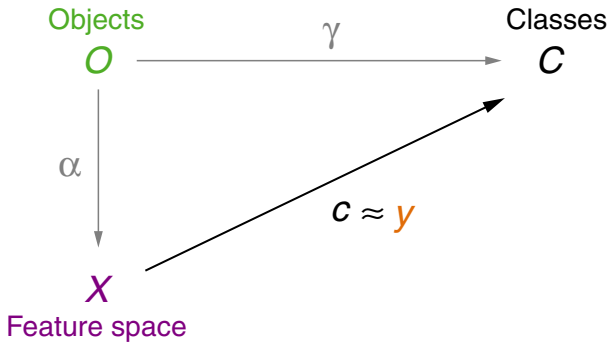
Exercise 1 : Machine Learning (general)

(a) Define the terms “supervised learning”, “unsupervised learning”, and “reinforcement learning”.

(b) Sketch for each learning paradigm a typical problem together with a description of its technical realization.

Exercise 2 : Specification of Learning Tasks

The following picture from the lecture slides describes the relationship between Real World and Model World, when it comes to the specification of learning tasks.



(a) Assume you are building a machine learning system that predicts whether a given mushroom is poisonous or edible. For the following list, decide which symbol from the picture most closely matches the given list item:

(a1) A pile of Mushrooms.

(a2) A table with the columns “size”, “weight”, and “color”, as well as one row for each mushroom, and the respective measurements in the cells.

(a3) A human mushroom expert who can tell whether any mushroom you show them is poisonous or edible.

(a4) A device that measures size, weight and color of a mushroom.

(a5) The set $\{\mathsf{Poisonous}, \mathsf{Edible}\}$

(a6) The machine learning system that you are trying to build.

(b) Look ahead to the programming exercise 6. When you complete task (6a) of this exercise, which symbol in the picture corresponds to the role that **you** are playing? Which symbol corresponds to the functions that you implement in (6b)?

Exercise 3 : Linear Regression

The table below describes four cars by their age and stopping distance for a full braking at 100km/h till stop:

Car	Wartburg	Moskvich	Lada	Trabi
Age (year)	5	7	15	28
Mileage (km)	30 530	90 000	159 899	270 564
Stopping distance (meter)	50	79	124	300

(a) Determine (by hand) the weights w_i for the linear regression for the age variable.

(b) Extrapolate the expected average stopping distance for the Lada car (i.e., age = 15 years). Use the model from (a).

(c) Consider the mileage of the cars as an additional variable and repeat (a) and (b) under this setting.

(d) Draw a scatter plot of the data points, and the linear regression for a variable of your choice (i.e., either age or mileage on the x-axis).

(e) Discuss the problems and pitfalls of extrapolation.

Exercise 4 : Concept Learning

The examples of a training set for a classification problem are described by the values of the attributes $A_1, \ldots, A_p$ and the related concepts $C = \{0, 1\}$. For the attributes $A_1, \ldots, A_p$ there are in each case $m_1, \ldots, m_p$ values, e.g. $a_{i,1}, \ldots, a_{i,m_i}$ for A_i. The hypothesis space contains the conjunctions of restrictions for the attributes: "A_1 has values a_{1,j_1} and ... and A_k has a_{k,j_k}". A question mark in a hypothesis denotes a wildcard for the respective attribute domain. The hypothesis space does also contain the empty hypothesis $\perp$, which assigns all examples to the concept 0.

(a) Determine the number $n(p)$ of all possible examples for this problem.

(b) Determine the number $|H_p|$ of different hypotheses.

(c) How will the above answers change, if an additional attribute A_{p+1} with m_{p+1} values is added? Derive a recursion formula.

Exercise 5 : Concept Learning (Background)

(a) The Find-S algorithm considers only positive training examples. Explain whether this property can cause the algorithm to return an inconsistent hypothesis. Assume the hypothesis setup from the lecture.

(b) Is the hypothesis constructed by the Find-S algorithm dependent on the example order? Explain your answer.

Exercise 6 : P Data Acquisition and Feature Engineering

In this course, we will use the Python programming language, version 3, to work with data sets and implement fundamental machine learning algorithms. Throughout the programming labs, we will work on the task of web page function classification: given an arbitrary web page, assign a class based on its content. This type of classifier is important, for example, for search engines to build better result pages. Concretely, we want to learn to distinguish pages that aim to...

- ... allow the visitor to discuss with others,
- ... suggest pages to the visitor,

- . . . to get information from the visitor,
- . . . to sell to or buy from the visitor,
- . . . to entertain the visitor,
- . . . to inform the visitor.

In brief, we will name the six classes **Discuss**, **Suggest**, **Collect**, **Trade**, **Entertain**, **Inform**.

The main purpose of this first programming exercise is to create a dataset that we will use throughout the following labs. To this end, each group will annotate several web pages with the correct class to create a ground truth, and implement a few simple features.

(a) The annotation tool accompanying this exercise will present you screen shots of web pages and offer a set of labels to choose from. Based on your group name (as selected in Moodle), you will be randomly assigned a set of examples to annotate. Open the tool in your browser, annotate all assigned examples and export the resulting data set. Upload the resulting CSV file in the assigned place in Moodle. Coordinate with your group members to divide up the work—note that if you do, you may have to upload multiple files. You'll find further instructions within the annotation tool itself.

(b) To have features for the classification, you then have to implement feature extraction functions that take as input a simple String containing the HTML of the webpage and that output a single number (e.g., an Integer or a Float). Later in the course, we will use these functions to extract numeric features per web page.

For now, your task is to implement any three features of your choice as Python functions. You can use the ideas from the following list as a guide, or come up with your own:

Basic:

- number of characters for inline scripts, inline styles or text content,
- number of HTML tags (total, or specific for metadata, content, forms, images, . . .),
- number of external resources (like scripts, styles, images),
- number of links (can be separated into absolute / relative, internal / external).

Intermediate:

- fraction of hyperlinked text (text content that describes the link vs. other text content),
- ratio of list elements per list (in menus, navs, `ul`, `ol`, . . .),
- number of tokens (like words, whitespaces, punctuation marks),
- number of word types (like nouns, verbs, adjective, stopwords, . . .),
- number of occurrences from filterlist (like restricted language, stopwords),
- DOM depth (*"how deeply nested are tags?"*, for all tags or specific tags, like `p`, `span`),
- ratio of children tags in parent tags,
- number of ads (often images, tags use specialized ids or CSS classes, maybe the URL?).

Note: Prefix each of your feature extraction function with `feature_`, followed by a descriptive name, like the following example:

```
def feature_rawlength(html):
    """Returns the raw text length of the ``html`` string."""
    return len(html)
```

Hint: Try to use the Python standard library where possible. For more complex ideas, you may use packages like lxml (for DOM parsing), parsel or beautifulsoup4 (for easy DOM element selection via CSS selectors). We will introduce some of these later on in the lab class.